Kapitola: Problémy so self-reference

Už sme videli viacero situácií, kedy možnosť odkazovať sa „na seba samého“ viedla k problémom: Russellov paradox holiča, nerozhodnuteľnosť problému zastavenia, nie-primitívna rekurzívnosť univerzálnej funkcie pre množinu primitívne rekurzívnych funkcií. (A ešte ich pár aj stretneme, napríklad Gödelovu vetu o neúplnosti, alebo hneď teraz jednu ďalšiu.)

1.1 Busy Beaver problem

Uvažujme deterministické Turingove stroje s nasledujúcimi vlastnosťami:

- pracovná abeceda je $\Gamma = \{0, 1\}$
- páska je obojsmerne nekonečná a na začiatku výpočtu je na každom políčku symbol 0
- v každom kroku výpočtu sa stroj musí pohnúť (doľava alebo doprava)
- stavy sú $\{0, 1, \ldots, n-1\} \cup \{H\}$ pre nejaké $n \in \mathbb{N}$
- stav 0 je začiatočný, dosiahnutím stavu H výpočet končí

Busy Beaver problém. K danému n nájsť ten stroj, ktorý má n stavov, spustený na prázdnom vstupe (samé 0) v konečnom čase zastane a v okamihu, keď zastane, bude na páske maximálny možný počet znakov 1.

Busy Beaver funkcia. Funkcia $B(n)$ udáva najväčší dosiahnuteľný počet jednotiek pre n stavov.

(Poznámka: Táto definícia je zjavne korektná. Totiž pre každé n existuje n-stavový stroj, ktorý na prázdnom vstupe zastane a tiež platí, že n-stavových strojov je len konečne veľa – presne $(4n+4)^{2n}$. Preto každé $B(n)$ je maximom nejakej konečnej neprázdnej množiny.)

1.2 Busy Beaver príklady

V nasledujúcej tabuľke je rekordér pre $n = 3$. Riadky sú čítané symboly, stĺpce stavy, v každom políčku je trojica (písaný symbol, smer pohybu hlavy, nový stav).

	0	1	2
0	1,→,1	0,→,2	1,←,2
1	1,→,H	1,→,1	1,←,0

Výpočet tohto stroja má 14 krokov. Keď zastane, na páske je 6 jednotiek. Platí teda $B(3) = 6$.

Je dokázané, že $B(4) = 11$. Pre $n = 5$ máme $24^{10} = 63\,403\,380\,965\,376$ rôznych TS. Spomedzi nich sa ešte o pár desiatkach nevie, či na prázdnom vstupe zastanú alebo nie. Doterajší rekordér vyrobí 4 098 jednotiek (a potrebuje na to približne 47 miliónov krokov).

A potom to už ide z kopca. Či presnejšie, do kopca. Pre $n = 6$ vyrobí najlepší známy TS viac ako 10^{1439} jednotiek v okamihu, keď zastane. (Tento TS bol nájdený v roku 2007.) Pre väčšie n sa už tiež vedia len nejaké dolné odhady. Napr. pre $n = 10$ je $B(10) > g(3^{27})$, kde $g(1) = 3$ a $g(n+1) = 3^{g(n)}$.

1.3 Bratranec Busy Beaver funkcie

Podobná funkcia ako $B(n)$ je nasledujúca funkcia $S(n)$: namiesto toho, koľko najviac jednotiek vieme vyrobiť pred tým, ako zastaneme, nás bude zaujímať, koľko najviac krokov môžeme spraviť.

Uvedomte si, že tieto dve maximá sa nemusia nadobúdať pre ten istý TS. Napr. pre $n = 3$ existuje TS, ktorý síce skončí s menej ako 6 jednotkami, ale zato spraví až 21 krokov.

Napriek tomu ale zjavne platí nerovnosť $B(n) \leq S(n)$: Ak vieme, že ľubovoľný n-stavový TS, ktorý zastane, zastane po nanajvýš $S(n)$ krokoch, tak zároveň vieme, že mohol vyrobiť nanajvýš $S(n)$ jednotiek – v každom kroku vie vyrobiť najviac jednu.

1.4 Nevypočítateľnosť funkcie S

Je pomerne jasné, že funkcia S je síce totálna, ale nie je rekurzívna. Môžeme to ukázať napr. redukciou na problém zastavenia na prázdnom vstupe.

Keby sme vedeli algoritmicky počítať S, vieme rozhodnúť problém zastavenia na prázdnom vstupe nasledovne: Na vstupe dostaneme TS A. Pozrieme sa, koľko má stavov, spočítame príslušnú hodnotu S, zväčšíme ju o 47 a následne odsimulujeme toľko krokov A. Ak počas simulácie nezastal, vieme, že už nezastane nikdy.

(Problém zastavenia na prázdnom vstupe je rovnako ťažký ako všeobecný problém zastavenia, každú inštanciu (TS,vstup) problému zastavenia vieme prerobiť na TS, ktorý na prázdnu pásku najskôr zapíše príslušný vstup a potom na ňom robí to isté, čo pôvodný TS.)

Tu môžeme urobiť ešte jedno zaujímavé pozorovanie: S nutne rastie rýchlejšie ako akákoľvek vypočítateľná funkcia. Keby sme totiž vedeli vypočítať hocijakú funkciu, ktorá rastie aspoň tak rýchlo ako S, vedeli by sme ju použiť na konštrukciu z predchádzajúcich odsekov.

Už Ackermannova funkcia rástla neuveriteľne rýchlo, rýchlejšie ako všetky primitívne rekurzívne funkcie, a už u primitívne rekurzívnych funkcií sme stretli neuveriteľne rýchlo rastúce funkcie. Funkcia S rastie neporovnateľne rýchlejšie ako všetky z nich.

1.5 Nevypočítateľnosť funkcie B

Sporom. Predpokladajme, že existuje TS *Bobor*, ktorý počíta $B(n)$ – keď ho spustíme na páske s n jednotkami, prerobí ju na pásku s $B(n)$ jednotkami a zastane.

(Problém bude samozrejme v tom, že TS *Bobor* má nejaký konkrétny počet stavov x, a zároveň má vedieť počítať funkciu B aj pre vstupy omnoho väčšie ako x.)

Ďalej v našom dôkaze uvažujme dva zjavne existujúce TS: $Pridaj1$, ktorý ide doľava kým nenájde nulu, a tú prepíše na jednotku, a $Dvakrat$, ktorý prerobí pásku s n jednotkami na pásku s $2n$ jednotkami.

Označme A celkový počet stavov týchto troch TS. Teraz vyrobíme TS $Pruser$ s $2A$ stavmi. Tento TS najskôr pomocou A stavov vyrobí A jednotiek na páske, potom zavolá $Dvakrat$, potom *Bobor* a nakoniec $Pridaj1$.

Zjavne tento stroj v konečnom čase zastane a vyrobí na páske $B(2A) + 1$ jednotiek, čo je spor.

1.6 Dá sa self-reference zbaviť?

Problém bol vždy schovaný v tom istom. Voľne podľa Scotta Aaronsona: Nič deterministické nikdy nemôže mať možnosť dokonalého sebapoznania – keby si vedel povedať, čo spravíš o desať sekúnd, mohol by si namiesto toho spraviť niečo iné.

In related news, práve vychádza prvá sezóna seriálu FlashForward, postaveného na veľmi podobnej premise: Celé ľudstvo na 137 sekúnd stratilo vedomie a počas tohto času každý videl svoju budúcnosť o pol roka. A hlavná otázka samozrejme je: stane sa naozaj o pol roka to, čo videli, alebo to práve vďaka tejto informácii dokážu zmeniť?[1]

vety o rekurzii http://www.cis.ksu.edu/ tamtoft/Papers/Amt+Nik+Tra+Jon:1989/short.pdf

1.7 Quine v Pythone

```
A='print chr(65)+chr(61)+chr(39)+A+chr(39)+chr(10)+A'
print chr(65)+chr(61)+chr(39)+A+chr(39)+chr(10)+A
```

[1] Je samozrejme jasné, ako to dopadne v seriáli, kvôli dramatickému efektu – naozaj sa stane to, čo videli, len to bude zasadené do nečakaného kontextu. Ale filozofická otázka je to pekná.